PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-182931
(43)Date of publication of application : 28.06.2002

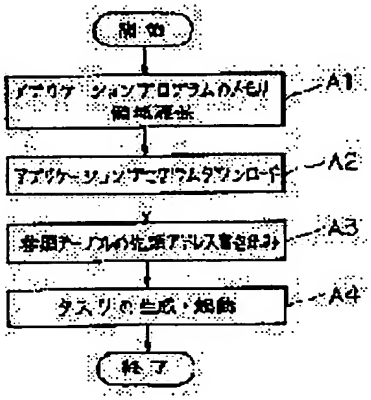
(51)Int.Cl. G06F 9/46

(21)Application number : 2000-384081
(22)Date of filing : 18.12.2000

(71)Applicant : YASKAWA ELECTRIC CORP
(72)Inventor : YUMIBA TADASUKE

(54) COMMON OS SYSTEM CALL METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To enable development of an application program not to depend on an OS.
SOLUTION: A common OS system call capable of selectively calling a system call of an operating system counted on an incorporating device is generated from system calls of several kinds of operating systems and a reference table to define staring addresses of the respective common OS system calls as an element is generated. The application program can indirectly call the system call of the operating system mounted on the incorporating device by calling the common OS system call by using the reference table.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（Ａ）　(11)特許出願公開番号
特開2002－182931
（P2002－182931A）
(43)公開日　平成14年６月28日(2002.6.28)

(51)Int.Cl.'	識別記号	ＦＩ	テーマコード'（参考）
Ｇ０６Ｆ　9/46	３４０	Ｇ０６Ｆ　9/46	３４０Ａ　５Ｂ０９８

審査請求　未請求　請求項の数３　ＯＬ　（全５頁）

(21)出願番号　特願2000－384081(P2000－384081)

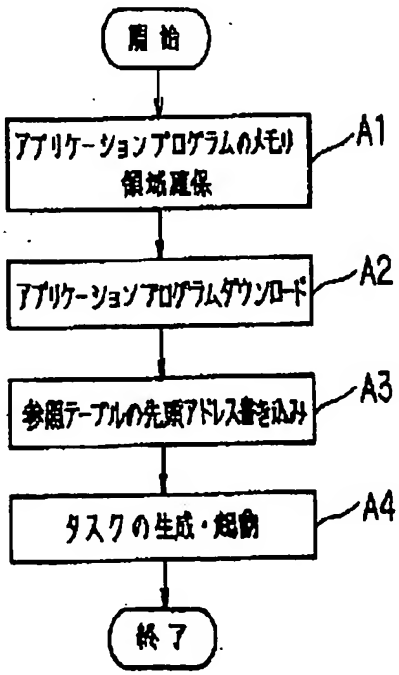
(22)出願日　平成12年12月18日(2000.12.18)

(71)出願人　000006622
株式会社安川電機
福岡県北九州市八幡西区黒崎城石２番１号
(72)発明者　弓場　忠輔
福岡県北九州市八幡西区黒崎城石２番１号
株式会社安川電機内
(74)代理人　100088328
弁理士　金田　暢之　（外２名）

Ｆターム(参考)　5B098 BA06 EE02 GA01 GA02 GB01

(54)【発明の名称】　共通ＯＳシステムコール方法

(57)【要約】
【課題】　ＯＳに依存しないアプリケーションプログラムの開発を可能とする。
【解決手段】　数種類のオペレーティングシステムのシステムコールの中から組み込み装置に実装されているオペレーティングシステムのシステムコールを選択的に呼び出し可能な共通ＯＳシステムコールを作成し、各共通ＯＳシステムコールの開始アドレスを要素とする参照テーブルを作成する。アプリケーションプログラムは、参照テーブルを用いて共通ＯＳシステムコールを呼び出すことによって、組み込み装置に実装されているオペレーティングシステムのシステムコールを間接的に呼び出すことができる。



【特許請求の範囲】
【請求項１】　数種類のオペレーティングシステムのシステムコールの中から実際に組み込み装置に実装されているオペレーティングシステムのシステムコールを選択的に呼び出し可能な共通ＯＳシステムコールを作成し、前記各共通ＯＳシステムコールの開始アドレスを要素とする参照テーブルを作成し、
前記参照テーブルを用いて前記共通ＯＳシステムコールを呼び出すことによって、前記組み込み装置に実装されているオペレーティングシステムのシステムコールを間接的に呼び出す共通ＯＳシステムコール方法。
【請求項２】　前記共通ＯＳシステムコールは、アプリケーションプログラムから直接呼び出される請求項１記載の共通ＯＳシステムコール方法。
【請求項３】　前記共通ＯＳシステムコールは、アプリケーションプログラムと前記オペレーティングシステムとの間のインタフェースであるアプリケーションプログラムインタフェースによって呼び出され、
前記アプリケーションプログラムは、前記アプリケーションプログラムインタフェースを呼び出すことによって前記共通ＯＳシステムコールの間接呼出しを行なう請求項１記載の共通ＯＳシステムコール方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、ダウンロードされたアプリケーションプログラムを実行する組み込み型装置に実装されたオペレーティングシステム（以下、ＯＳ）のシステムコールを呼び出すための共通ＯＳシステムコール方法に関する。
【０００２】
【従来の技術】従来、ダウンロードされたアプリケーションプログラムインタフェース（以下、ＡＰＩ）を使用しないアプリケーションプログラムを実行する組み込み型装置では、アプリケーションプログラムが直接、ＯＳシステムコールを呼び出していた。
【０００３】図５には、ＯＳシステムコールの一例が示されている。これらのＯＳシステムコールは、Ｃ言語で記載されている。図５（ａ）の関数ＧｅｔＭｅｍｏｒｙ（）と図５（ｂ）の関数ＡｌｌｏｃＭｅｍｏｒｙ（）とは、それぞれＯＳ＿ＡとＯＳ＿Ｂとにおけるメモリを確保するためのシステムコールである。
【０００４】図５に示すように、ＯＳが異なると、システムコールの関数名および引数は異なったものになる。そのため、従来の組み込み装置では、ＯＳが修正あるいは変更された場合に、システムコールを呼び出すアプリケーションプログラムのソースコードを書き直さなければならなくなるという問題があった。
【０００５】また、直接システムコールを呼び出すようなアプリケーションプログラムを開発するには、ＯＳ等のシステムプログラムに関する深い知識が必要となるため、アプリケーションプログラムの開発が容易でないという問題もあった。
【０００６】
【発明が解決しようとする課題】以上述べたように、従来の組み込み型装置では、以下に示す２つの問題点を有している。
（１）　ＯＳが修正あるいは変更された場合に、システムコールを呼び出すアプリケーションプログラムのソースコードを書き直さなければならなくなる。
（２）　アプリケーションプログラムの開発においても、ＯＳ等のシステムプログラムに関する深い知識が必要となり、アプリケーションプログラムの開発が容易でない。
【０００７】本発明は、ＯＳに依存しないアプリケーションプログラムの開発を可能とする共通ＯＳシステムコール方法を提供することを目的とする。
【０００８】また、本発明は、容易にアプリケーションプログラムの開発することができる共通ＯＳシステムコール方法を提供することを目的とする。
【０００９】
【課題を解決するための手段】上記課題を解決するために、本発明の共通ＯＳシステムコール方法では、数種類のオペレーティングシステムのシステムコールの中から実際に組み込み装置に実装されているオペレーティングシステムのシステムコールを選択的に呼び出し可能な共通ＯＳシステムコールを作成し、前記各共通ＯＳシステムコールの開始アドレスを要素とする参照テーブルを作成し、前記参照テーブルを用いて前記共通ＯＳシステムコールを呼び出すことによって、前記組み込み装置に実装されているオペレーティングシステムのシステムコールを間接的に呼び出す。
【００１０】本発明の共通ＯＳシステムコール方法では、いずれのオペレーティングシステムが組み込み型装置に実装されていても、そのオペレーティングシステムのシステムコールを呼び出し可能な共通ＯＳシステムコールを用いてアプリケーションプログラムの開発を行なうことができるため、ＯＳに依存しないアプリケーションプログラムの開発が可能となる。
【００１１】本発明の他の共通ＯＳシステムコール方法では、前記共通ＯＳシステムコールは、アプリケーションプログラムと前記オペレーティングシステムとの間のインタフェースであるアプリケーションプログラムインタフェースによって呼び出され、前記アプリケーションプログラムは、前記アプリケーションプログラムインタフェースを呼び出すことによって前記共通ＯＳシステムコールの間接呼出しを行なう。
【００１２】本発明の共通ＯＳシステムコール方法では、アプリケーションプログラムインタフェースが共通ＯＳシステムコールの呼出しを行なうので、アプリケーションプログラムの開発者は、ＯＳ等のシステムプログ

ラムに関する深い知識を持たなくてもアプリケーションの開発を行なうことができるようになるため、容易にアプリケーションプログラムの開発することができるようになる。

【００１３】

【発明の実施の形態】次に、本発明の一実施形態の共通ＯＳシステムコール方法について図面を参照して詳細に説明する。本実施形態の共通ＯＳシステムコール方法では、ＯＳがＯＳ＿Ａであっても、ＯＳ＿Ｂであっても呼び出し可能な共通ＯＳシステムコールとしての関数を用意する。図１には、本実施形態の共通ＯＳシステムコール方法における共通ＯＳシステムコールの一例が示されている。

【００１４】図１の関数ｙｅ＿ｇｅｔ＿ｍｅｍ（）は、ＯＳ＿ＡおよびＯＳ＿Ｂに共通なシステムコールである。この関数ｙｅ＿ｇｅｔ＿ｍｅｍ（）では、もし、“ＯＳ＿Ａ”が真であれば、ＯＳ＿ＡのシステムコールｇｅｔＭｅｍｏｒｙ（）が実行され、“ＯＳ＿Ａ”が偽であればＯＳ＿ＢのシステムコールａｌｌｏｃＭｅｍｏｒｙ（）が実行される。つまり、共通ＯＳシステムコールは、数種類のＯＳのシステムコールの中から実際に組み込み装置に実装されているＯＳのシステムコールを選択的に呼び出し可能となっている。このような共通ＯＳシステムコールを用意すれば、従来のように、ＯＳを変更する毎にアプリケーションプログラムのソースコードも変更する必要がなくなる。

【００１５】本実施形態の共通ＯＳシステムコール方法では、上述のような共通ＯＳシステムコールを数十個用意し、図２に示す参照テーブルを作成する。この参照テーブルの要素は各共通ＯＳシステムコールの開始アドレス、すなわち各共通ＯＳシステムコールへのポインタとなっている。

【００１６】図３は、本実施形態の共通ＯＳシステムコール方法を用いて作成されたアプリケーションプログラムが組み込み装置のメモリにダウンロードされた際のメモリ配置を示す図である。アプリケーションプログラムのメモリ配置は、ヘッダセクション１と、スタートアップ関数２と、テキストセクション３と、データセクション４とから構成される。

【００１７】ヘッダセクション１の固定オフセットアドレスには、前述の参照テーブルが書き込まれる。スタートアップ関数２はアプリケーションプログラム生成時にアプリケーションプログラムにリンクされる関数であり、その処理の先頭アドレスは、アプリケーションプログラムの実行が開始されるときの実行開始アドレスとなっており、その実行が開始されるときに最初に実行される関数である。

【００１８】スタートアップ関数２は、アプリケーションプログラムのタスクの起動や、組み込み装置のローカルメモリへの参照テーブルの読み出し等の処理を行なう。テキストセクション３には、アプリケーションプログラムのコードが格納され、データセクション４には、アプリケーションプログラムのデータが格納される。

【００１９】このアプリケーションプログラムにおいて、共通ＯＳシステムコールが呼ばれると、組み込み装置のＣＰＵは、まず、ヘッダセクション１の固定オフセットアドレスに格納されている参照テーブルへのポインタに基づいて参照テーブルへアクセスし、参照テーブルに格納されている呼び出された共通ＯＳシステムコールへのポインタに基づいて、共通ＯＳシステムコールにジャンプする。

【００２０】図４は、本実施形態の共通ＯＳシステムコール方法におけるアプリケーションプログラムをダウンロードする際の動作を示すフローチャートである。まず、アプリケーションプログラムを書き込むのに必要な領域を、組み込み型装置内部のメモリから確保し（ステップＡ１）、確保された領域にアプリケーションプログラムをダウンロードする（ステップＡ２）。次に、図５に示すヘッダセクション１内の固定オフセットアドレスに、参照テーブルへのポインタとして参照テーブルの先頭アドレスを書き込む（ステップＡ３）。そして、スタートアップ関数２を実行してタスクの生成および起動を行なう（ステップＡ４）。

【００２１】以上述べたように、本実施形態の共通ＯＳシステムコール方法では、いずれのオペレーティングシステムが前記組み込み型装置に実装されていても、当該オペレーティングシステムのシステムコールを呼び出し可能な共通ＯＳシステムコールを用いてアプリケーションプログラムの開発を行なうことができるため、ＯＳに依存しないアプリケーションプログラムの開発が可能となる。

【００２２】また、本実施形態の共通ＯＳシステムコール方法では、アプリケーションプログラムが直接、共通ＯＳシステムコールの呼出しを行なったが、本発明の共通ＯＳシステムコール方法はこれに限定されるものではなく、アプリケーションプログラムが直接、共通ＯＳシステムコールの呼出しを行なわずに、アプリケーションプログラムインタフェース（以下、ＡＰＩ）が共通ＯＳシステムコールの呼出しを行なうようにしてもよい。図６にＡＰＩが共通ＯＳシステムコールの呼出しを行う場合の一例を示す。図６（ａ）はアプリケーションのソースコードであり、図６（ｂ）はＡＰＩのソースコードである。図６（ａ）に示されるアプリケーションプログラムは、ＡＰＩの関数、例えば、図６（ｂ）のｙｅＡｌｌｏｃＭｅｍｏｒｙ（）を読み出すことによって組み込み型装置に対する指令を実行する。ＡＰＩの関数、例えばｙｅＡｌｌｏｃＭｅｍｏｒｙ（）は、アプリケーションプログラムからの指令に応じ、必要に応じて共通ＯＳシステムコールの間接呼び出しを行なう。

【００２３】この場合、アプリケーションプログラムイ

ンタフェースが共通ＯＳシステムコールの呼出しを行なうので、アプリケーションプログラムの開発者は、ＯＳ等のシステムプログラムに関する深い知識を持たなくてもアプリケーションの開発を行なうことができるようになるため、容易にアプリケーションプログラムの開発することができるようになる。

【００２４】

【発明の効果】以上述べたように、本発明の共通ＯＳシステムコール方法は、以下に示す２つの利点を有する。

（１）　いずれのオペレーティングシステムが組み込み型装置に実装されていても、当該オペレーティングシステムのシステムコールを呼び出し可能な共通ＯＳシステムコールを用いてアプリケーションプログラムの開発を行なうことができるため、ＯＳに依存しないアプリケーションプログラムの開発が可能となる。

（２）　アプリケーションプログラムインタフェースが共通ＯＳシステムコールの呼出しを行なうので、アプリケーションプログラムの開発者は、ＯＳ等のシステムプログラムに関する深い知識を持たなくてもアプリケーションプログラムの開発を行なうことができるようになるため、容易にアプリケーションプログラムの開発することができるようになる。

【図面の簡単な説明】

【図１】本発明の一実施形態の共通ＯＳシステムコール方法における共通ＯＳシステムコールの一例を示す図である。

【図２】本発明の一実施形態の共通ＯＳシステムコール方法における参照テーブルの一例を示す図である。

【図３】本発明の一実施形態の共通ＯＳシステムコール方法を用いて作成されたアプリケーションプログラムが組み込み装置のメモリにダウンロードされた際のメモリ配置を示す図である。

【図４】本発明の一実施形態の共通ＯＳシステムコール方法におけるアプリケーションプログラムをダウンロードする際の動作を示すフローチャートである。

【図５】ＯＳシステムコールの一例を示す図である。

【図６】ＡＰＩが共通ＯＳシステムコールの呼出しを行なう場合の一例を示す図である。

【符号の説明】

１　　ヘッダセクション

２　　スタートアップ関数

３　　テキストセクション

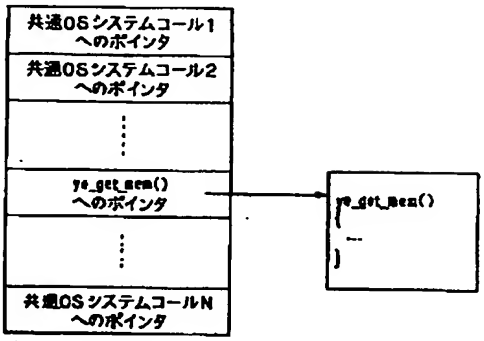
４　　データセクション

Ａ１～Ａ４　　ステップ

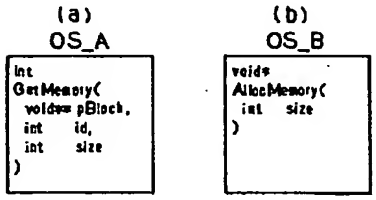
【図１】

```
int
ye_get_mem(
  void** pBlock,
  int    size,
)
{
  #if OS_A
    GetMemory();
  #elif
    AllocMemory();
  #endif
    return;
}
```

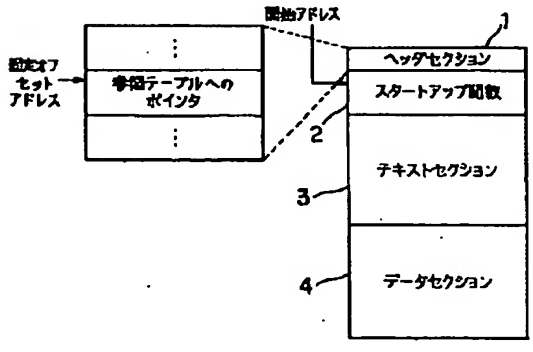
【図２】



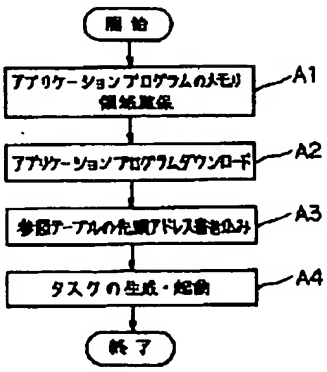
【図５】



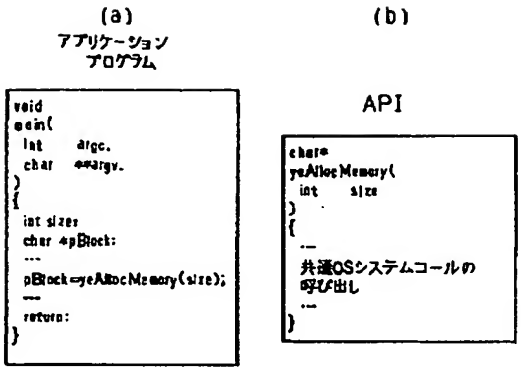
【図３】



【図４】



【図６】


(a)
アプリケーション
プログラム
void
main(
int argc,
char **argv,
)
{
int size;
char *pBlock;
---
pBlock=yeAllocMemory(size);
---
return;
}
(b)
API
char*
yeAllocMemory(
int size
)
{
--
共通OSシステムコールの
呼び出し
--
}